Project Proposal

Big Data Technologies - CSP 554

Comparative Analysis of Cloud-Based NoSQL Databases

Team Members

Urjita Saxena - A20578349

Harsh Sharma - A20563703

Nikita Sharma – A20588827

Trushaliben Chandulal Tanti – A20598441

1. Introduction and Motivation

As applications increasingly demand real-time, large-scale data processing, distributed NoSQL databases have become vital for scalability and flexibility. Unlike traditional relational databases, NoSQL systems provide horizontal scaling, schema flexibility, and fault tolerance.

This project compares **AWS DynamoDB**, **MongoDB Atlas**, and **Apache Cassandra** to evaluate how their architectures influence performance, scalability, and cost efficiency. Each represents a distinct data model key-value/document, document, and wide-columnvand consistency approach, allowing an empirical study of CAP theorem trade-offs in practice.

2. Objectives

1. Benchmark CRUD and query latency under identical workloads.
2. Measure throughput scalability.
3. Compare cost trends between on-demand and provisioned modes.
4. Examine replication, partitioning, and consistency mechanisms.
5. Recommend database choices for use cases like IoT, e-commerce, and analytics.

3. Background

Traditional relational databases often struggle to handle the scalability, flexibility, and high-throughput demands of modern applications. NoSQL databases address these challenges by efficiently managing large volumes of unstructured or semi-structured data.

However, NoSQL systems differ significantly in design. MongoDB (document store), Cassandra (wide-column store), and DynamoDB (key-value store) each balance consistency, latency, and fault tolerance differently.This project provides an **empirical comparison** of these databases to highlight their strengths, limitations, and most suitable use cases across varied cloud workloads.

4. Methodology

Dataset & Environment

- Dataset: Public IoT or e-commerce dataset from *Kaggle*.
- MongoDB Atlas: Free-tier M10 cluster (sharded).
- AWS DynamoDB: Two tables provisioned vs on-demand.
- Cassandra: Deployed via *DataStax Astra DB* or Docker.

Benchmarking
Python scripts using pymongo, boto3, and cassandra-driver will perform sequential and concurrent CRUD tests. Metrics:

- Latency (ms), throughput (ops/sec), and resource usage.
  Monitoring via CloudWatch, Atlas Metrics, and Astra Insights.

Analysis

- Perform dataset profiling (mean, std, nulls).
- Plot latency vs throughput and cost vs workload size.
- Evaluate consistency and scaling behavior.

5. Expected Results

- **DynamoDB:** Lowest latency, predictable scaling.
- **MongoDB:** Balanced performance, strong query flexibility.
- **Cassandra:** Highest write scalability, near-linear horizontal scaling.

Overall, DynamoDB is expected to excel in managed scalability, MongoDB in query flexibility, and Cassandra in distributed writes. Charts will summarize latency, throughput, and cost per operation.

6. Ethical and Operational Considerations

Only public, anonymized datasets will be used. API keys will be secured and deleted after testing. Cloud resources will be shut down post-experiment to minimize cost and energy use. Scripts and configurations will be fully documented for reproducibility.

7. References

1. Amazon Web Services. (2024). *Amazon DynamoDB Developer Guide.* Retrieved from https://docs.aws.amazon.com/dynamodb/
2. MongoDB Inc. (2024). *MongoDB Atlas Documentation.* Retrieved from https://www.mongodb.com/docs/
3. DataStax. (2024). *Cassandra Architecture Overview.* Retrieved from https://www.datastax.com/resources/whitepapers/
4. Han, J., Haihong, E., Le, G., & Du, J. (2011). *Survey on NoSQL Databases. Journal of Cloud Computing.* Retrieved from https://journalofcloudcomputing.springeropen.com/articles/10.1186/2192-113X-2-1